



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

پروژه پنجم (گزارش کد) هوش مصنوعی

رشته علوم کامپیوتر

گزارش کار پروژه پیش‌بینی برچسب درآمد بالای ۵۰ هزار دلار

نگارش

محمد رضا شیخ الاسلامی

استاد درس

دکتر مهدی قطعی

استاد کارگاه

آقای بهنام یوسفی مهر

۱۴۰۳ بهمن

## چکیده

این پروژه با هدف پیش‌بینی سطح درآمد افراد (بیشتر یا کمتر از ۵۰ هزار دلار در سال) با استفاده از الگوریتم‌های یادگیری ماشین انجام شده است. دیتاست مورد استفاده مجموعه داده Adult Income از مخزن UCI است که شامل ویژگی‌های جمعیت‌شناسختی افراد مختلف می‌باشد. پس از انجام تحلیل اکتشافی و پیش‌پردازش داده‌ها، مدل‌هایی نظیر Logistic Regression ، K-Nearest Neighbors ، MLP و SVM ، XGBoost ، LightGBM ، Random Forest .Decision Tree ارزیابی شدند و نتایج نشان داد مدل XGBoost بهترین عملکرد را داشته و می‌تواند با دقت حدود ۸۶٪ سطح درآمد افراد را پیش‌بینی کند.

## واژه‌های کلیدی:

، Random Forest .Decision Tree Logistic Regression ، K-Nearest Neighbors  
MLP ، SVM ، XGBoost ، LightGBM

صفحه

فهرست مطالب

۱	چکیده
۳	فصل اول مقدمه
۵	فصل دوم شرح مراحل انجام شده در پروژه
۹	فصل سوم : پاسخ به سؤالات داخل تمپلیت
۳۶	فصل چهارم تحلیل داده ها در EDA
۳۸	فصل پنجم توضیحات درباره ی پیش پردازش های انجام شده
۴۱	فصل ششم عملکرد مدل های استفاده شده
۴۵	فصل هفتم مقایسه نتایج به دست آمده
۵۰	منابع و مراجع

## فصل اول مقدمه

در سال‌های اخیر، استفاده از یادگیری ماشین برای تحلیل داده‌های جمعیت‌شناختی و اجتماعی گسترش فراوانی یافته است. یکی از مسائل مطرح در این زمینه، پیش‌بینی سطح درآمد افراد بر اساس ویژگی‌های فردی، شغلی و خانوادگی است. این نوع پیش‌بینی‌ها می‌توانند در حوزه‌هایی مانند بازاریابی هدفمند، تخصیص منابع، ارزیابی اعتباری و سیاست‌گذاری‌های دولتی بسیار مفید واقع شوند و این پروژه با هدف طراحی و ارزیابی مدل‌های یادگیری ماشین برای طبقه‌بندی سطح درآمد افراد به دو دسته‌ی کمتر یا مساوی ۵۰ هزار دلار ( $K \geq 50$ ) و بیشتر از ۵۰ هزار دلار ( $K < 50$ ) در سال انجام شده است. داده‌های مورد استفاده از پایگاه داده UCI (University of California Irvine) گرفته شده‌اند و شامل اطلاعات بیش از ۴۸ هزار فرد مختلف هستند.

## توضیح در مورد دیتاست

دیتاست Adult Income از اطلاعات سرشماری سال‌های گذشته تشکیل شده و دارای ۱۴ ویژگی (ستون) ورودی و یک ستون خروجی (برچسب) است. در ادامه شرح کوتاهی از هر ستون آورده شده است:

نام ستون	نوع داده	توضیح
age	عددی	سن فرد
workclass	طبقه‌ای	نوع سازمان یا نهاد کاری (خصوصی، دولتی، خوداشتغال و...)
fnlwgt	عددی	وزن نمونه (برای تعیین داده‌های آماری)
education	طبقه‌ای	(...) و سطح تحصیلات (Bachelors, Masters, 11th)
education-num	عددی	مقدار عددی معادل سطح تحصیلات
marital-status	طبقه‌ای	وضعیت تأهل (متاهل، مجرد، طلاق‌گرفته و...)
occupation	طبقه‌ای	عنوان شغلی (مدیر، کارگر، فروشنده، معلم و...)
relationship	طبقه‌ای	نسبت خانوادگی با سرپرست خانواده (همسر، فرزند، خود فرد و...)
race	طبقه‌ای	نژاد (سفیدپوست، سیاهپوست، آسیایی و...)
sex	طبقه‌ای	جنسیت (مرد، زن)
capital-gain	عددی	میزان سود سرمایه‌ای کسب شده
capital-loss	عددی	میزان زیان سرمایه‌ای
hours-per-week	عددی	تعداد ساعت‌های کاری در هفته
native-country	طبقه‌ای	کشور محل تولد فرد
income (برچسب)	طبقه‌ای	این همان هدف مدل است - ( $<=50K$ یا $>50K$ ) سطح درآمد

## فصل دوم

### شرح مراحل انجام شده در پروژه

در این پروژه مراحل مختلفی برای آماده‌سازی داده‌ها، ساخت مدل‌ها و تحلیل نتایج طی شده‌اند. در ادامه هر مرحله به صورت مختصر شرح داده شده است:

#### 1. وارد کردن کتابخانه‌های مورد نیاز

در اولین قدم، کتابخانه‌های لازم برای تحلیل داده، پیش‌پردازش، مدل‌سازی و رسم نمودار وارد شدند. که از مهم‌ترین کتابخانه‌ها می‌توان به موارد زیر اشاره کرد :

- **pandas** برای خواندن و مدیریت داده‌های جدولی (DataFrame)
- **numpy** برای انجام محاسبات عددی
- **seaborn** و **matplotlib** برای مصورسازی داده‌ها
- **sklearn** برای الگوریتم‌های یادگیری ماشین و ابزارهای ارزیابی

#### 2. بارگذاری داده‌ها

دو فایل داده‌ای `adult.data` و `adult.test` خوانده شدند که این فایل‌ها شامل داده‌های آموزشی و تست هستند و با توجه به اینکه در فایل تست برچسب درآمد به صورت `50K>(با نقطه)` ذخیره شده است ، باید در گام‌های بعدی این مورد اصلاح شود.

### 3. تمیزسازی برچسبها در داده‌های آزمون

در فایل adult.test، مقادیر ستون درآمد به صورت  $K50=$  و  $>K50$  ذخیره شده بودند که با استفاده از متدها str.replace() نقطه از انتهای آنها حذف شد تا با داده‌های آموزشی همخوانی داشته باشند

### 4. بررسی اولیه داده‌ها

برای آشنایی با ساختار دیتابیس از توابع زیر استفاده شده است :

- برای نمایش چند ردیف اول داده Head()
- برای بررسی تعداد ردیفها و ستون‌ها shape
- برای بررسی نوع داده‌ها و وجود مقادیر گمشده info()

همچنین فرآنی برچسب‌های درآمد نیز بررسی شد تا از متعادل بودن کلاس‌ها اطلاع حاصل شود.

### 5. تحلیل آماری اولیه و بصری (EDA)

در این مرحله داده‌ها با روش‌های زیر تحلیل و مصورسازی شدند:

- بررسی مقادیر گمشده و نامعتبر
- هیستوگرام برای ویژگی‌های عددی مانند age و hours-per-week
- نمودار میله‌ای برای ویژگی‌های طبقه‌ای مانند education, workclass, occupation
- رسم نمودار Boxplot برای مقایسه ویژگی‌ها بین گروه‌های درآمدی
- بررسی نسبت تعداد داده‌ها در هر کلاس ( $K50=$  و  $>K50$ )
- رسم heatmap برای نمایش همبستگی ویژگی‌های عددی
- بررسی رابطه ویژگی‌ها با متغیر هدف (درآمد)

## 6. پیشپردازش داده‌ها

برای آماده‌سازی داده‌ها جهت مدل‌سازی، پیش‌پردازش‌هایی انجام شد:

- حذف ردیف‌هایی که مقدار ? داشتند (مثالاً در occupation)
- پر کردن مقادیر گم شده با مقدار Missing
- استفاده از LabelEncoder برای تبدیل مقادیر متغیر به عددی در مدل‌های درختی
- استفاده از Logistic Regression برای مدل OneHotEncoder
- مقیاس‌بندی ویژگی‌های عددی با StandardScaler

## 7. بالانس کردن کلاس‌ها (Class Balancing)

در مسئله‌ی پیش‌بینی درآمد، متغیر هدف income شامل دو کلاس است:

K50=> (درآمد کمتر یا مساوی ۵۰ هزار دلار)

K50< (درآمد بالای ۵۰ هزار دلار)

که با بررسی توزیع کلاس‌ها مشاهده شد که داده‌ها نامتوازن (imbalanced) هستند، به این معنا که تعداد نمونه‌های کلاس K50=> به مراتب بیشتر از K50< است. این نامتوازنی می‌تواند باعث شود مدل یادگیری تمایل به پیش‌بینی کلاس غالب (یعنی K50=>) داشته باشد و کلاس اقلیت را نادیده بگیرد.

## 7. آموزش مدل‌ها

هشت مدل یادگیری ماشین آموزش داده شدند:

- :KNN .1
- :Decision Tree .2
- :Random Forest .3
- :XGBoost .4
- :LightGBM .5
- Logistic Regression .6
- :SVM .7
- Multilayer Perceptron (MLP .8

## 8. پیش‌بینی و ارزیابی مدل‌ها

با استفاده از داده‌های تست، برای هر مدل پیش‌بینی انجام شد و معیارهای زیر محاسبه شدند:

- Accuracy
- Precision
- Recall
- F1-score
- ROC Curve + AUC
- ماتریس سردرگمی (Confusion Matrix)

از این ارزیابی‌ها برای مقایسه عملکرد مدل‌ها استفاده شده است.

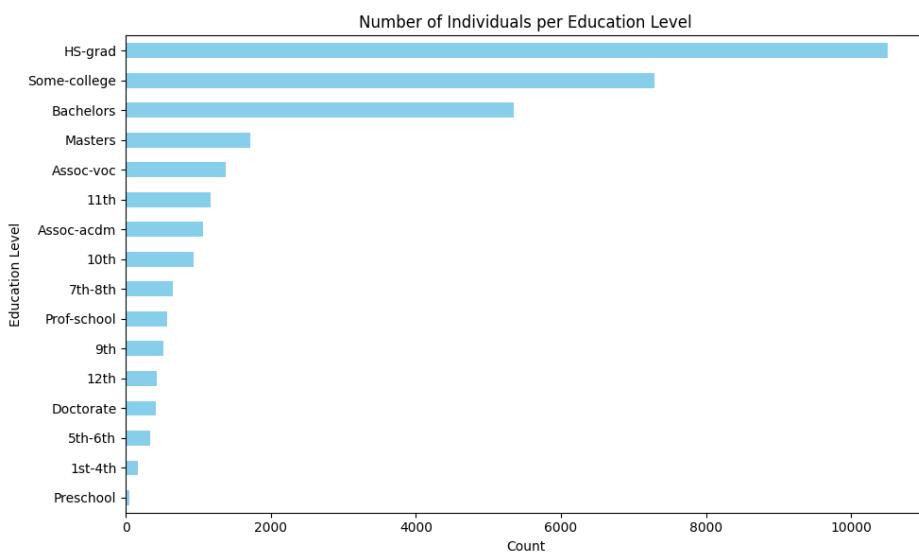
## 9. مقایسه عملکرد مدل‌ها

در پایان، عملکرد مدل‌ها در قالب جدول مقایسه شد تا مشخص شود کدام مدل در طبقه‌بندی درآمد بهترین نتیجه را داشته است

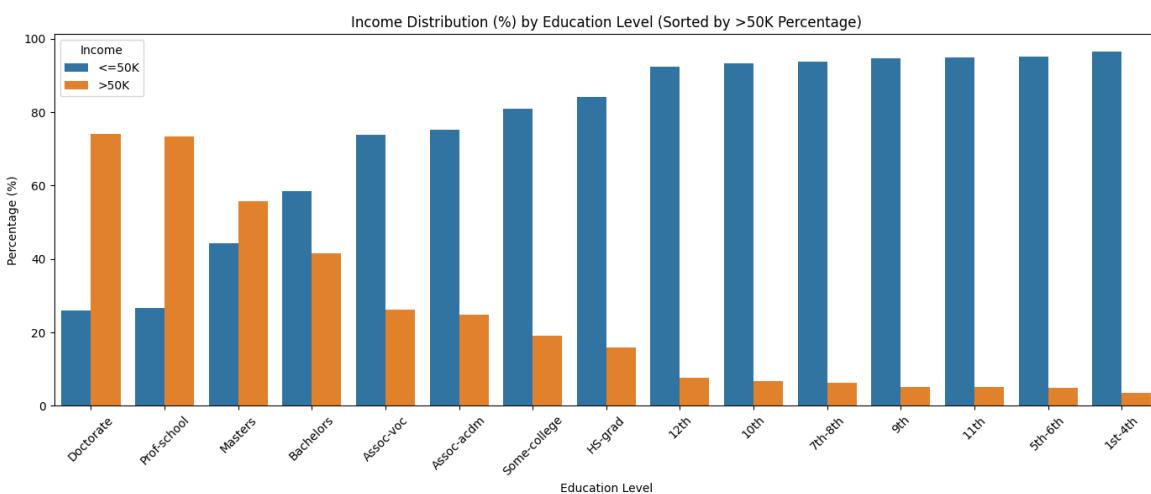
## فصل سوم : پاسخ به سؤالات داخل تمپلیت

**سؤال اول :** ۱- بیشترین مدرک تحصیلی کسب شده در بین شرکت کنندگان چیست؟

۲- آیا ارتباطی بین سطح تحصیلات و درآمد وجود دارد؟



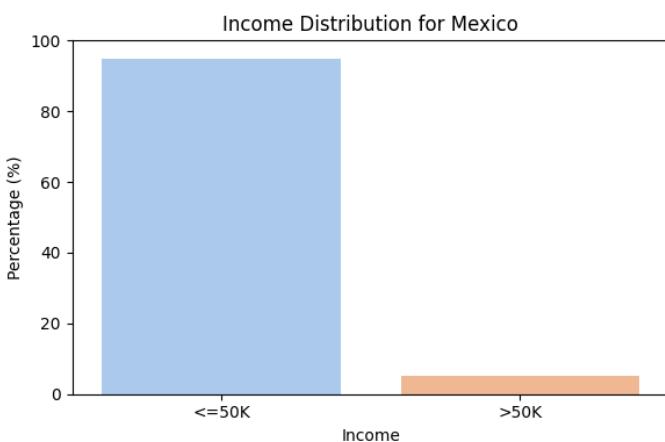
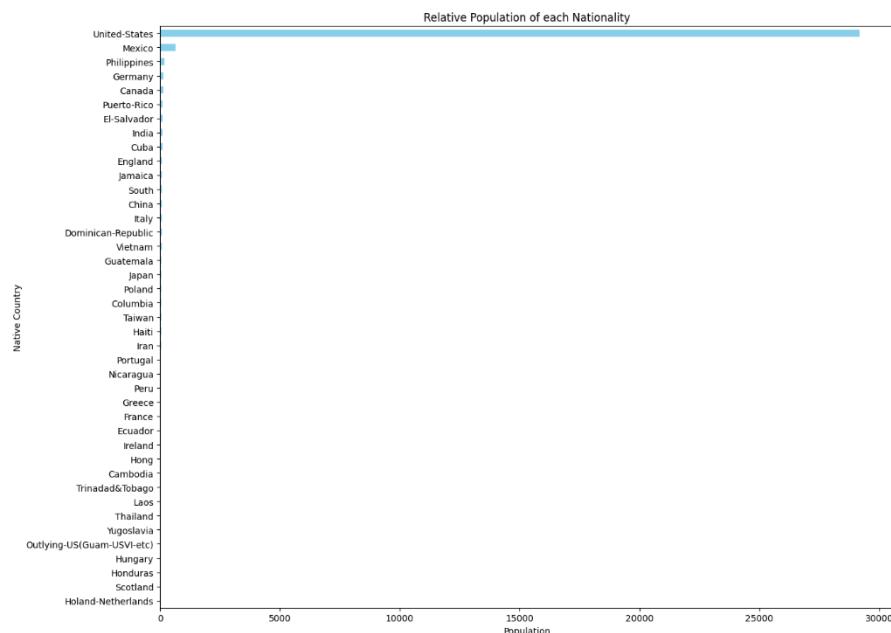
بیشترین مدرک تحصیلی کسب شده در بین شرکت کنندگان، **HS-grad** است.



سه سطح تحصیلی دکترا، پروفسوری و کارشناسی ارشد بیشترین درصد درآمد بالای ۵۰ هزار دلار را دارند، بنابراین با افزایش سطح تحصیلات، درصد درآمد بالای ۵۰ هزار دلار نیز بیشتر می‌شود.

## سوال دوم : ۱- کدام ملیت دومین کشور پرجمعیت آمریکا است و توزیع درآمد آنها چگونه است؟ و

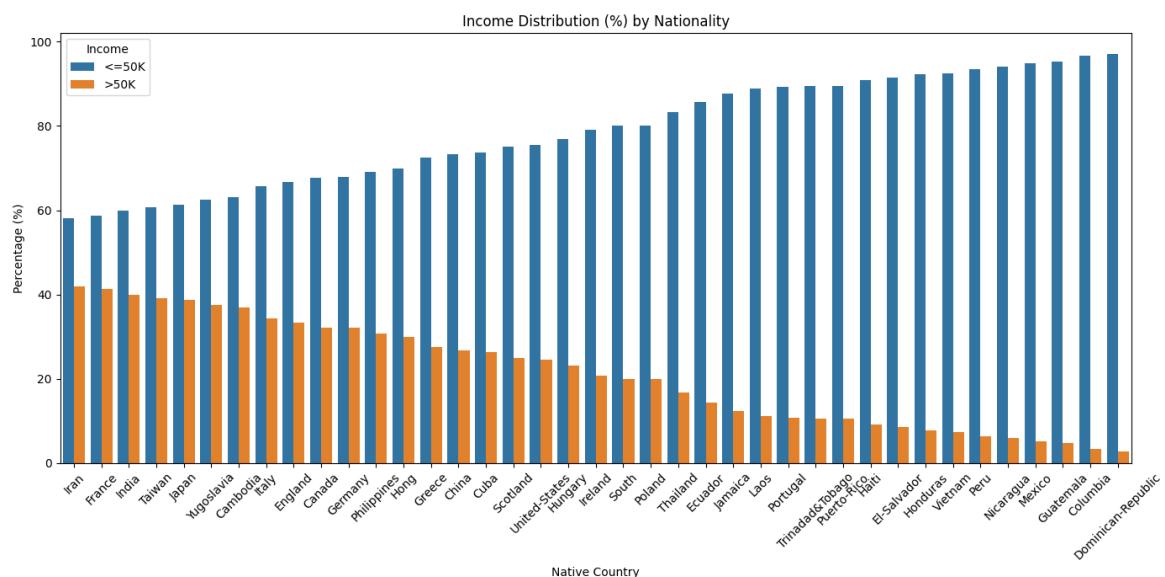
چه عواملی ممکن است این الگوهای درآمدی را توضیح دهد؟



بر اساس نمودار میله‌ای رسم شده، مکزیک دومین کشور پرجمعیت آمریکا است و توزیع درآمد آن نشان می‌دهد که حدود ۹۰٪ از جمعیت آن کمتر از ۵۰ هزار دلار درآمد دارند. عواملی مانند ملیت، ساعت کاری و تحصیلات می‌توانند این الگوی درآمدی را توضیح دهند.

## سوال سوم : ۱- کدام ملیت بیشترین نسبت افراد با درآمد بیش از ۵۰ هزار دلار را دارد؟ میتوانید

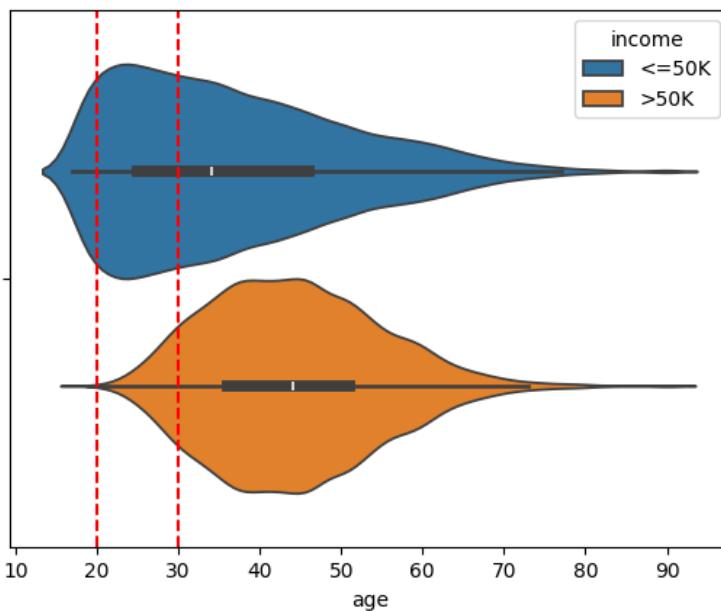
توضیح دهید؟



ایران بالاترین نسبت افرادی را دارد که بیش از ۵۰ هزار دلار درآمد دارند (حدود ۴۲٪ از جمعیت آن)

## سوال چهارم : ۱- چه الگوهایی را می‌توان در گروه سنی با درآمد بالای ۵۰ هزار دلار مشاهده کرد؟

۲- آیا داشتن درآمد بالای ۵۰ هزار دلار برای افراد جوان‌تر رایج است؟



### الگوهای گروه سنی با درآمد بالای ۵۰ هزار:

**تراکم بالاتر در میانسالی:** نمودار ویولونی برای گروه درآمدی «بالای ۵۰ هزار» تراکم بالاتری (بخش وسیع‌تر) را بین سنین تقریباً ۳۵ تا ۵۰ سال نشان می‌دهد. این نشان می‌دهد که افراد در این محدوده سنی احتمال بیشتری دارد که درآمدی بالای ۵۰ هزار داشته باشند.

**کاهش تراکم با افزایش سن:** با افزایش سن به بالای ۵۰ هزار، تراکم نمودار ویولونی برای درآمد «بالای ۵۰ هزار» به تدریج کاهش می‌یابد. این نشان می‌دهد که در حالی که افراد بالای ۵۰ سال هنوز می‌توانند درآمدی بالای ۵۰ هزار داشته باشند اما احتمال آن با افزایش سن کاهش می‌یابد.

**میانگین سنی:** به نظر می‌رسد میانگین سنی ( نقطه سفید ) برای گروه درآمدی «بالای ۵۰ هزار» حدود ۴۵ سال باشد. این موضوع بیشتر از این مشاهده پشتیبانی می‌کند که افراد میانسال احتمال بیشتری دارد که درآمدی بالای ۵۰ هزار داشته باشند.

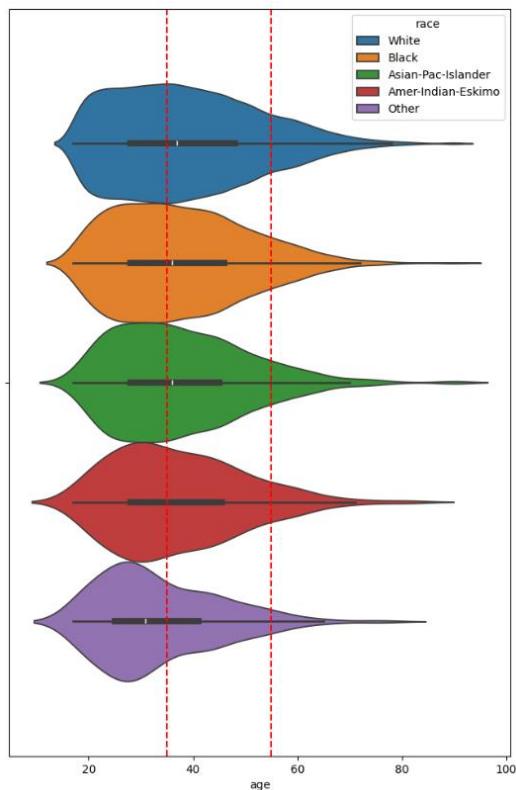
### آیا داشتن درآمد بالای ۵۰ هزار برای افراد جوان تر رایج است؟

بر اساس نمودار ویولونی، داشتن درآمد بالای ۵۰ هزار برای افراد جوان تر کمتر رایج است. که از دلایل آن می توان به موارد زیر اشاره کرد:

تراکم کمتر در سنین پایین تر: نمودار ویولونی برای گروه درآمدی «بالای ۵۰ هزار» برای افراد زیر ۳۰ سال نسبتاً باریک است. این نشان دهنده تراکم کمتر نقاط داده است، که نشان می دهد افراد جوان کمتری درآمد بالای ۵۰ هزار دارند.

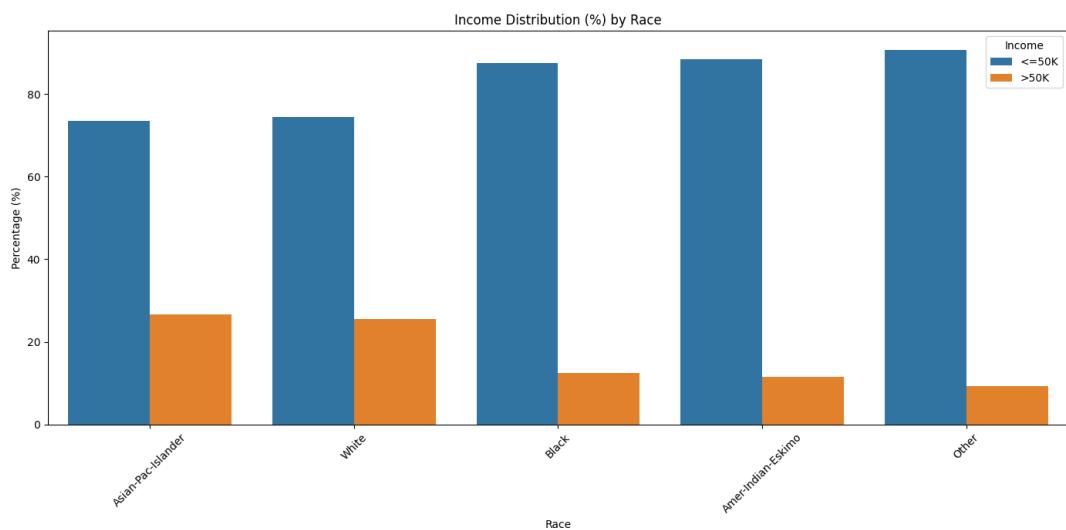
چولگی: نمودار ویولونی برای گروه درآمدی «بالای ۵۰ هزار» به نظر می رسد که به راست متمايل است، به اين معنى که دنباله بيشتر به سمت سنین بالاتر امتداد مي يابد. اين موضوع بيشتر تأييد مي کند که درآمد بالاتر در بين افراد مسن تر شائع تر است.

## سوال پنجم : ۱- کدام نژاد بیشترین تراکم افراد میانسال را دارد؟



بر اساس نمودار ویولونی ، نژاد سفید بیشترین تمرکز افراد میانسال را دارد.

## سوال ششم : ۱- کدام نژاد بیشترین نسبت افراد با درآمد بالا را دارد؟



نژاد آسیایی-اقیانوسیه-جزایری و سفیدپوست بالاترین نسبت افراد با درآمد بالا را دارند

**سوال هفتم :** بر اساس بینش‌های بخش‌های قبلی، شخصیتی را توصیف کنید که احتمال بالای برای کسب درآمد بالای ۵۰ هزار دلار در سال داشته باشد. عواملی مانند سن، نژاد، وضعیت تأهل و وضعیت تأهل را برای تشریح ویژگی‌های این فرد در نظر بگیرید.

فردی با ویژگی‌های زیر شناس بالایی برای کسب درآمد بالای ۵۰ هزار دلار در سال دارد:

سن: میانسال (۴۰-۵۰)

مدرک تحصیلی: دکترا

کشور محل تولد: ایران

نژاد: آسیایی-جزیره‌نشین اقیانوس آرام

وضعیت تأهل: متأهل-شهروند-همسر

طبقه شغلی: خوداشتغال

جنسیت: مرد

نسبت: همسر یا شوهر

شغل: مدیر اجرایی

**سوال هشتم :** به نظر شما کدام روش(ها) برای این مجموعه داده بهترین هستند؟

چرا؟ تعداد مقادیر از دست رفته، نوع ویژگی‌ها و چگونگی تأثیر آن بر مدل‌سازی را در نظر بگیرید.

من مقادیر تهی را با یک نام دسته جدید مانند "Missing" جایگزین می‌کنم زیرا هر سه ستون Occupation و workclass از نوع طبقه‌بندی هستند و جایگزینی مقادیر گمشده با یک برچسب جدید مانند "Missing" به مدل اجازه می‌دهد تا یاد بگیرد که آیا خود فقدان قابل پیش‌بینی است یا خیر و از آنجایی که Occupation و workclass بیش از ۵٪ فقدان دارند، پر کردن آنها با mode می‌تواند توزیع ویژگی را مختل کند.

**سوال نهم :** روش‌های بالا را بررسی کنید (یا بر اساس آنچه در کلاس بحث کردہ‌ایم).

برای هر روش، به طور خلاصه موارد زیر را بنویسید:

چه زمانی ممکن است از آن استفاده کنید و مزایا و معایب بالقوه آن روش (روش‌هایی) را که فکر می‌کنید برای این مجموعه داده بهترین عملکرد را دارند، انتخاب کنید و توضیح دهید که چرا انتخاب شما مناسب است.

این روش برای هر دسته منحصر به فرد در یک ویژگی، یک ستون دودویی جدید ایجاد می‌کند و فقط یکی از ستون‌ها برای هر مشاهده با علامت "1" مشخص می‌شود و بقیه "0" هستند. همچنین تمام اطلاعات دسته را بدون اعمال هیچ ترتیبی حفظ می‌کند. با این حال، اگر دسته‌ها زیاد باشند، می‌تواند تعداد ویژگی‌ها را تا حد زیادی افزایش دهد.

☞ چه زمانی استفاده شود:

وقتی تعداد دسته‌های منحصر به فرد کم است.

به خصوص برای مدل‌های مبتنی بر درخت (جنگل تصادفی، XGBoost) خوب است.

مزایا: 

هیچ ترتیبی را در دسته‌ها فرض نمی‌کند.

استقلال دسته را حفظ می‌کند.

معایب: 

ابعاد را افزایش می‌دهد (ستون‌های جدید زیادی).

وقتی ستون مقادیر منحصر به فرد زیادی دارد (مثلاً بیش از 50 کشور) مناسب نیست.

هر دسته به یک مقدار صحیح منحصر به فرد تبدیل می‌شود.

 چه زمانی استفاده شود:

وقتی دسته‌ها ترتیب طبیعی دارند (مثلاً کم < متوسط < زیاد).

برای مدل‌های مبتنی بر درخت حتی بدون ترتیب هم قابل قبول است.

مزایا: 

ساده و کارآمد ( فقط یک ستون).

با مدل‌های مبتنی بر درخت که به فواصل عددی متکی نیستند، به خوبی کار می‌کند.

معایب: 

در صورت عدم دقیق، یک ترتیب مصنوعی را القا می‌کند (برای مدل‌های خطی نامناسب است).

می‌تواند مدل‌های مبتنی بر فاصله مانند KNN یا SVM را گیج کند.

**Frequency Encoding**: هر دسته را با فرکانس آن (تعداد رخدادها در مجموعه داده) جایگزین می‌کند.

چه زمانی استفاده شود:

وقتی می خواهید رمزگذاری را فشرده نگه دارید و اهمیت دسته را ثبت کنید.

با هر دو مدل درختی و خطی کار می کند.

**مزایا:**

فشر ۵ (تک ستونی).

اطلاعات مفیدی (محبوبیت یک دسته) را ثبت می‌کند.

معايم X

هویت دسته‌های جداگانه را از دست می‌دهد.

در صورت نشست اطلاعات دسته‌ها، می‌تواند منجر به پیش‌برازش شود.

**Target / Mean Encoding**: هر دسته با مقدار میانگین متغیر هدف برای آن دسته جایگزین می‌شود.

چه زمانی استفاده شود:

وقتی، یک متغیر هدف دارید (مانند سطح درآمد).

با ویژگی‌های با کار دینا لیستی بالا به خوبی کار می‌کند.

من ایا:

می‌تواند با کدگذاری مستقیم تأثیر دسته، عملکرد را بهبود بخشد.

 معایب:

خطر نشت هدف در صورت عدم انجام با اعتبارسنجی متقابل.  
اگر مجموعه داده‌ها کوچک یا نامتعادل باشد، می‌تواند بیش‌برازش شود.

**Binary Encoding:** دسته‌ها به صورت اعداد صحیح کدگذاری می‌شوند، سپس آن اعداد صحیح به فرمت دودویی تبدیل می‌شوند. هر رقم دودویی به یک ستون ویژگی جداگانه تبدیل می‌شود و تعداد کل ستون‌ها را کاهش می‌دهد.

 چه زمانی استفاده شود:

وقتی تعداد دسته‌ها متوسط تا زیاد است (مثلاً ۵۰-۱۵).  
جایگزین خوبی برای One-Hot وقتی می‌خواهید در حافظه صرفه‌جویی کنید.

 مزایا:

ابعاد کم.

بیش‌برازش کمتری نسبت به کدگذاری هدف دارد.

 معایب:

تفسیر دشوار.

در ستون‌های با تعداد هسته کوچک، نسبت به الگوریتم وان-هات (one-hot) اثربخشی کمتری دارد.

هر دسته با استفاده از یک تابع هش به تعداد ثابتی از ستون‌ها تبدیل می‌شود.

چه زمانی استفاده شود:

ویژگی‌های با تعداد هسته بسیار بالا (مثلاً بیش از ۱۰۰ دسته). وقتی تفسیرپذیری برایتان مهم نیست.

مزایا:

بسیار فشرده و سریع.  
نیازی به ذخیره نگاشت دسته‌ها نیست.

معایب:

خطر تصادم هش.  
قابل تفسیر نیست.

Column	Unique Values	Best Encoding	Why?
workclass	~9	One-Hot	Low cardinality; categories independent
education	~16	Label Encoding	Ordered (if grouped), but One-Hot is safer
marital-status	~7	One-Hot	Clear categories;
occupation	~15	Label Encoding	Medium cardinality; save space
relationship	~6	One-Hot	Low cardinality
race	~5	One-Hot	Low cardinality
sex	2	Binary (0/1)	Simplest case
native-country	~42	Label Encoding	High cardinality; one-hot too large
income	2	Binary (0/1)	Target variable, already binary

## سوال دهم : یک روش مقیاس‌بندی انتخاب کنید و دلیل خود را توضیح دهید

من از Standard Scaler برای مقیاس‌بندی ویژگی‌ها استفاده کردم، چون چند ویژگی عددی داریم و بیشتر ویژگی‌های عددی توزیع نرمال دارند، بنابراین بهتر است از Standard Scaler استفاده کنیم.

## سوال یازدهم : توزیع کلاس را بررسی کنید و یک استراتژی نمونه‌گیری مجدد انتخاب کنید و دلیل خود را توضیح دهید.

در این صورت، آیا باید از نمونه‌گیری مجدد استفاده کنیم یا نه، و چرا؟

استراتژی انتخاب شده: **SMOTE** (تکنیک بیش‌نمونه‌گیری مصنوعی اقلیت) استدلال:

**Random Oversampling**: ممکن است با تکرار نقاط داده واقعی، بیش‌برازش ایجاد کند.

**Random Undersampling**: داده‌های ارزشمند را از کلاس اکثربیت حذف می‌کند.

با درونیابی بین نمونه‌های نزدیک، نمونه‌های جدید مصنوعی از کلاس اقلیت ایجاد می‌کند - که به مدل کمک می‌کند تا بهتر تعمیم یابد.

بنابراین، SMOTE معمولاً زمانی که داده‌های کافی دارید و می‌خواهید توزیع کامل کلاس اکثربیت را حفظ کنید، نسبت به نمونه‌گیری مجدد اولیه ترجیح داده می‌شود.

بله، زیرا تعداد افرادی که درآمد کمتر از ۵۰ هزار دلار دارند بسیار بیشتر از تعداد افرادی است که درآمد بیشتر از ۵۰ هزار دلار دارند.

## سوال دوازدهم : چگونه «K» در KNN بر عملکرد مدل تأثیر می‌گذارد؟

«K» در K-نزدیک‌ترین همسایه‌ها (KNN) تعداد همسایه‌هایی را که هنگام پیش‌بینی در نظر گرفته می‌شوند، کنترل می‌کند و تأثیر قابل توجهی بر موازنی بایاس-واریانس و عملکرد مدل دارد:

### K کوچک (مثلاً K=1 یا 3):

- بایاس کم، واریانس بالا.
- مدل بسیار انعطاف‌پذیر است و می‌تواند الگوهای پیچیده را ثبت کند.
- با این حال، به نویز حساس است و ممکن است داده‌های آموزشی را بیش‌برازش کند. پیش‌بینی‌ها بر اساس تعداد بسیار کمی از نقاط نزدیک انجام می‌شوند که می‌تواند منجر به نتایج ناپایدار شود.

### K بزرگ (مثلاً K=15 یا 50):

- بایاس زیاد، واریانس کم.
- مدل پایدارتر و روان‌تر می‌شود.
- بهتر تعییم می‌دهد اما ممکن است با نادیده گرفتن الگوهای محلی، کمتر برآرازش داشته باشد.
- داده‌های پرت تأثیر کمتری دارند، اما ممکن است روندهای محلی مهم از دست بروند.

K بسیار بزرگ (به اندازه مجموعه داده‌ها نزدیک می‌شود): مدل ممکن است فقط کلاس اکثریت (در طبقه‌بندی) یا میانگین همه نقاط (در رگرسیون) را پیش‌بینی کند و قدرت پیش‌بینی را کاهش دهد.

## سوال سیزدهم : یک درخت تصمیم چگونه تصمیم می‌گیرد که داده‌ها را در چه زمانی تقسیم کند؟

یک درخت تصمیم با انتخاب ویژگی و آستانه‌ای که به بهترین شکل داده‌ها را بر اساس متغیر هدف به گروههای مجزا تقسیم می‌کند، تصمیم می‌گیرد که داده‌ها را به کجا تقسیم کند. این کار را با موارد زیر انجام می‌دهد:

**ارزیابی ویژگی‌ها و آستانه‌ها:** برای هر ویژگی، الگوریتم نقاط تقسیم ممکن را در نظر می‌گیرد (به عنوان مثال، برای ویژگی‌های عددی، ممکن است مقادیر بین نقاط داده را آزمایش کند؛ برای ویژگی‌های دسته‌بندی، زیرمجموعه‌هایی از دسته‌ها را آزمایش می‌کند).

**اندازه‌گیری ناخالصی:** برای هر تقسیم بالقوه، یک معیار ناخالصی محاسبه می‌کند، معمولاً:

**ناخالصی جینی:** احتمال طبقه‌بندی نادرست یک عنصر تصادفی انتخاب شده را اندازه‌گیری می‌کند (هرچه کمتر بهتر باشد).

**آنتروپی/بهره اطلاعات:** کاهش عدم قطعیت در مورد متغیر هدف را پس از تقسیم اندازه‌گیری می‌کند (بهره بالاتر بهتر است).

**کاهش واریانس:** برای رگرسیون استفاده می‌شود و واریانس متغیر هدف را در گره‌های فرزند به حداقل می‌رساند.

**انتخاب بهترین تقسیم:** الگوریتم ویژگی و آستانه‌ای را انتخاب می‌کند که ناخالصی را در گره‌های فرزند حاصل به حداقل می‌رساند (یا بهره اطلاعات را به حداقل می‌رساند).

این فرآیند برای هر گره فرزند تا زمانی که یک معیار توقف برآورده شود (مثلاً حداقل عمق، حداقل نمونه در هر گره یا عدم بهبود بیشتر) تکرار می‌شود. هدف، ایجاد تقسیم‌هایی است که همگن‌ترین گره‌های فرزند را با توجه به متغیر هدف تولید می‌کنند و قدرت پیش‌بینی درخت را بهبود می‌بخشند.

## سوال چهاردهم : چرا جنگل تصادفی از بیش از یک درخت تصمیم استفاده می کند؟

جنگل تصادفی از چندین درخت تصمیم‌گیری برای بهبود دقت پیش‌بینی و کاهش بیش‌برازش استفاده می‌کند. دلیل آن این است:

کاهش واریانس: درخت‌های تصمیم‌گیری منفرد مستعد بیش‌برازش هستند، زیرا می‌توانند نویز موجود در داده‌ها را ثبت کنند. جنگل تصادفی با میانگین‌گیری از پیش‌بینی‌های چندین درخت (روش گروهی)، واریانس را کاهش می‌دهد و منجر به نتایج پایدارتر و قابل تعمیم‌تر می‌شود.

تنوع از طریق تصادفی‌سازی: هر درخت روی یک زیرمجموعه تصادفی از داده‌ها (از طریق بوت‌استرپ) و یک زیرمجموعه تصادفی از ویژگی‌ها در هر تقسیم آموزش داده می‌شود. این تنوع تضمین می‌کند که درختان الگوهای مختلفی را ثبت می‌کنند، همبستگی بین آنها را کاهش می‌دهند و عملکرد کلی را بهبود می‌بخشند.

مقاومت: ترکیب چندین درخت باعث می‌شود مدل نسبت به داده‌های پرت یا نویز در هر درخت واحد حساسیت کمتری داشته باشد، زیرا خطاهای هنگام تجمعی پیش‌بینی‌ها (مثلاً از طریق رأی‌گیری اکثریت برای طبقه‌بندی یا میانگین‌گیری برای رگرسیون) تمایل به حذف شدن دارند.

تعمیم‌پذیری بهتر: پیش‌بینی‌های جمعی بسیاری از درختان معمولاً دقیق‌تر از هر درخت واحد هستند، به خصوص در مجموعه داده‌های پیچیده.

## سوال پانزدهم : چرا XGBoost درخت‌ها را به ترتیب می‌سازد؟

XGBoost درخت‌ها را به ترتیب می‌سازد تا با تمرکز بر اصلاح خطاهای درخت‌های قبلی، مدل را به صورت تکراری بهبود بخشد. دلیل آن این است:

**چارچوب تقویت گراییان:** XGBoost مبتنی بر تقویت گراییان است، که در آن هر درخت برای پیش‌بینی باقیمانده‌ها (خطاهای) گروه قبلى آموزش داده می‌شود. با اضافه کردن متوالی درخت‌ها، مدل به تدریج خطای کلی را کاهش می‌دهد و یکتابع زیان (مثلاً میانگین مربعات خطای رگرسیون یا لگاریتم زیان برای طبقه‌بندی) را بهینه می‌کند.

**اصلاح خطای:** هر درخت جدید برای رفع نقاط ضعف مدل فعلی با برآذش به گراییان منفی تابع زیان ساخته می‌شود. این به XGBoost اجازه می‌دهد تا بر نمونه‌های طبقه‌بندی نشده یا پیش‌بینی نشده تمرکز کند و با هر مرحله دقت را بهبود بخشد.

**یادگیری افزایشی:** پیش‌بینی‌های همه درخت‌ها به صورت افزایشی ترکیب می‌شوند و هر درخت یک بهبود وزنی در پیش‌بینی نهایی ایجاد می‌کند. ساخت متوالی تضمین می‌کند که درخت‌های بعدی به جای شروع از ابتدا، مدل را اصلاح می‌کنند.

**منظم‌سازی و بهینه‌سازی:** XGBoost از منظم‌سازی (مثلاً جریمه‌های L1/L2) استفاده می‌کند و از اطلاعات گراییان مرتبه دوم (Hessian) برای بهروزرسانی‌های دقیق استفاده می‌کند که از ساخت درخت متوالی برای تنظیم دقیق مدل به طور مؤثر بهره می‌برد.

با ساخت درخت‌ها به ترتیب، XGBoost یک یادگیرنده قوی از یادگیرنده‌های ضعیف (درخت‌ها) ایجاد می‌کند و به دقت و استحکام بالایی، به ویژه در مجموعه داده‌های پیچیده، دست می‌یابد.

## سوال شانزدهم: چگونه LightGBM درختان را به طور متفاوتی گسترش می‌دهد؟

**LightGBM** درختان را به صورت برگ به برگ (best-first) گسترش می‌دهد و برگ را با بیشترین کاهش زیان تقسیم می‌کند و برخلاف روش‌های سنتی که درختان را به صورت سطح به سطح رشد می‌دهند که این موضوع باعث می‌شود سریع‌تر و دقیق‌تر عمل کند، اما در صورت عدم کنترل، می‌تواند منجر به بیش‌برازش شود. همچنین از تقسیم مبتنی بر هیستوگرام برای سرعت بخشیدن به آموزش و کاهش استفاده از حافظه استفاده می‌کند.

## سوال هفدهم : فرمول خطی در این رگرسیون لجستیک چگونه استفاده می‌شود؟

در رگرسیون لجستیک، از فرمول خطی برای مدل‌سازی رابطه بین ویژگی‌های ورودی و لگاریتم شанс نتیجه استفاده می‌شود که سپس برای پیش‌بینی احتمالات تبدیل می‌شود. نحوه کار آن به شرح زیر است:

1. ترکیب خطی: ویژگی‌های ورودی  $x_1, x_2, \dots, x_n$  به صورت خطی با وزن‌ها (ضرایب) ترکیب می‌شوند که وزن‌ها شامل  $\beta_1, \beta_2, \dots, \beta_n$  می‌شوند و فرمول ترکیب خطی آن برابر عبارت مقابل است :

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

که  $\beta_0$  همان عرض از مبدأ و  $Z$  همان ترکیب خطی است .

2. تبدیل سیگموئید: برای تبدیل لگاریتم شанс (z) به احتمال (p) (بین 0 و 1)، تابع سیگموئید اعمال می‌شود که

$$p = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}}$$

که این، ترکیب خطی (z) را به یک احتمال نگاشت می‌کند.

3. پیش‌بینی: برای طبقه‌بندی دودویی، یک آستانه (اغلب 0.5) بر (p) اعمال می‌شود: که اگر ( $p > 0.5$ ) باشد کلاس 1 را پیش‌بینی می‌کند و اگر ( $p < 0.5$ ) کلاس 0 را پیش‌بینی می‌کند

## سوال هجدهم : چگونه SVM داده‌هایی را که نمی‌توان با یک خط مستقیم از هم جدا کرد،

مدیریت می‌کند؟

وقتی داده‌ها را نمی‌توان با یک خط مستقیم از هم جدا کرد (یعنی به صورت خطی قابل جدا شدن نیستند)، ماشین‌های بردار پشتیبان (SVM) از دو استراتژی اصلی برای مدیریت این موضوع استفاده می‌کنند:

ترفند هسته : این قدرتمندترین و رایج‌ترین روش است و SVM از یک تابع هسته برای تبدیل داده‌های اصلی به فضایی با ابعاد بالاتر استفاده می‌کند که در آن یک جداکننده خطی می‌تواند وجود داشته باشد و این تبدیل به صورت ضمنی انجام می‌شود، به این معنی که SVM در واقع مختصات جدید را محاسبه نمی‌کند - فقط حاصلضرب‌های نقطه‌ای را در فضای جدید به طور موثر محاسبه می‌کند.

کرنل‌های محبوب:

RBF (تابع پایه شعاعی) / گاووسی - عالی برای الگوهای پیچیده.

چندجمله‌ای - تعاملات بین ویژگی‌ها را ثبت می‌کند.

سیگموئید - مانند فعال‌سازی شبکه عصبی رفتار می‌کند.

مثال: اگر داده‌ها در فضای دو بعدی مانند یک دایره به نظر برسند و نتوان آنها را به صورت خطی جدا کرد، یک هسته می‌تواند آن را به فضای سه بعدی منتقل کند که در آن یک صفحه مسطح می‌تواند کلاس‌ها را از هم جدا کند.

حاشیه نرم (متغیرهای Slack) : حتی پس از اعمال یک هسته، جداسازی کامل ممکن است به دلایل زیر امکان‌پذیر نباشد:

1 . نویز

2 . کلاس‌های همپوشانی

3 . SVM با معرفی یک حاشیه نرم، که توسط پارامتر C کنترل می‌شود، امکان برخی از طبقه‌بندی‌های نادرست را فراهم می‌کند:

C پایین → امکان طبقه‌بندی نادرست بیشتر (تمییم بهتر) را فراهم می‌کند.

C بالا → جداسازی دقیق را اجباری می‌کند (خطر بیش‌برازش).

## سوال نوزدهم : در طول فرآیند یادگیری در یک شبکه عصبی چه اتفاقی می‌افتد؟

در طول فرآیند یادگیری در یک شبکه عصبی، مدل به صورت تکراری پارامترهای خود را تنظیم می‌کند تا خطای پیش‌بینی‌هایش را به حداقل برساند. این فرآیند که به عنوان آموزش شناخته می‌شود، شامل مراحل کلیدی زیر است:

**مقداردهی اولیه:** وزن‌ها و بایاس‌ها (پارامترهای) شبکه، معمولاً با مقادیر تصادفی کوچک یا با استفاده از تکنیک‌های مقداردهی اولیه خاص مقداردهی اولیه می‌شوند تا یادگیری پایدار تضمین شود.

**انتشار رو به جلو:** داده‌های ورودی از طریق شبکه تغذیه می‌شوند و از لایه‌های نورون‌ها عبور می‌کند. هر نورون مجموع وزنی ورودی‌های خود را محاسبه می‌کند، یک بایاس اضافه می‌کند و یک تابع فعال‌سازی (مثلًا ReLU، سیگموئید یا  $\tanh$ ) را برای معرفی غیرخطی بودن اعمال می‌کند.

**خروجی لایه نهایی:** پیش‌بینی مدل است (مثلًا احتمالات کلاس برای طبقه‌بندی یا یک مقدار پیوسته برای رگرسیون).

**محاسبه زیان:** تفاوت بین خروجی پیش‌بینی‌شده و هدف واقعی با استفاده از یک تابع زیان (مثلًا میانگین مربعات خطا برای رگرسیون، آنتروپی متقاطع برای طبقه‌بندی) اندازه‌گیری می‌شود. زیان، میزان فاصله‌ی پیش‌بینی‌های مدل از مقادیر واقعی را اندازه‌گیری می‌کند.

**بازتاب پسانشان:** خطا (زیان) در شبکه به عقب منتشر می‌شود تا گرادیان‌های تابع زیان را با توجه به هر وزن و بایاس محاسبه کند و این کار با استفاده از قانون زنجیره‌ای برای تعیین میزان سهم هر پارامتر در خطا انجام می‌شود.

**بهروزرسانی پارامتر (بهینه‌سازی):** یک الگوریتم بهینه‌سازی، معمولاً گرادیان نزولی وزن‌ها و بایاس‌ها را در جهتی که زیان را کاهش می‌دهد، بهروزرسانی می‌کند.

قانون بهروزرسانی عبارت است از:  $\text{وزن} = \text{نرخ یادگیری} * \text{گرادیان} - \text{وزن}$  ، که در آن نرخ یادگیری، اندازه‌ی گام بهروزرسانی‌ها را کنترل می‌کند.

این مرحله، شبکه را برای پیش‌بینی‌های بهتر تنظیم می‌کند.

تکرار: این فرآیند (انتشار رو به جلو، محاسبه تلفات، انتشار معکوس و بهروزرسانی پارامتر) برای چندین دوره (گذر کامل از مجموعه داده‌های آموزشی) تکرار می‌شود و هر دوره، داده‌ها را در بخش‌های کوچک‌تری به نام دسته‌ها پردازش می‌کند تا بین کارایی محاسباتی و بهروزرسانی‌های پایدار تعادل برقرار شود.

## سوال نوزدهم : آیا Grid Search برای فضاهای جستجوی بزرگ کارآمد است؟

Grid Search به دلیل ماهیت جامع خود برای فضاهای جستجوی بزرگ کارآمد نیست. دلیل آن شامل موارد زیر است :

**شمارش جامع:** جستجوی شبکه‌ای هر ترکیب ممکن از ابرپارامترها را در شبکه مشخص شده ارزیابی می‌کند. این امر به صورت نمایی با تعداد پارامترها و مقادیر افزایش می‌یابد و آن را از نظر محاسباتی برای فضاهای جستجوی بزرگ گران می‌کند.

**هزینه محاسباتی:** هر ترکیب نیاز به آموزش و ارزیابی مدل (اغلب با اعتبارسنجی متقلبل) دارد که می‌تواند زمان بر باشد، به خصوص برای مدل‌های پیچیده (به عنوان مثال، شبکه‌های عصبی) یا مجموعه داده‌های بزرگ و برای فضاهای جستجوی بزرگ، این می‌تواند ساعتها، روزها یا بیشتر طول بکشد.

**بازده نزولی:** در فضاهای جستجوی بزرگ، بسیاری از ترکیبات ابرپارامترها ممکن است عملکرد مشابهی داشته باشند، اما جستجوی شبکه‌ای همه آنها را صرف نظر از پیکربندی‌های کمی متفاوت ارزیابی می‌کند و منابع را هدر می‌دهد.

**عدم انطباق:** جستجوی شبکه‌ای بر اساس نتایج میانی سازگار نمی‌شود و کورکورانه همه ترکیبات را بدون اولویت‌بندی مناطق امیدوارکننده فضای جستجو آزمایش می‌کند.

## سوال بیستم : تفاوت اصلی Grid Search با Randomized Search چیست؟

تفاوت اصلی بین جستجوی تصادفی و جستجوی شبکه‌ای در رویکرد آنها برای کاوش در فضای ابرپارامتر نهفته است:

جستجوی شبکه‌ای:

جستجوی جامع: تمام ترکیبات ممکن از ابرپارامترهای مشخص شده در یک شبکه از پیش تعریف شده را ارزیابی می‌کند.

مثال: برای 3 پارامتر با 5 مقدار برای هر کدام، ترکیب‌های  $5 * 5 * 5 = 125$  را آزمایش می‌کند.

مزایا: یافتن بهترین ترکیب در داخل شبکه را تضمین می‌کند.

معایب: از نظر محاسباتی پرهزینه است، به خصوص برای فضاهای جستجوی بزرگ، زیرا تعداد ترکیب‌ها به صورت نمایی افزایش می‌یابد.

جستجوی تصادفی:

نمونه‌گیری تصادفی: تعداد ثابتی از ترکیب‌های تصادفی ( $n_{iter}$ ) را از فضای پارامتر نمونه‌برداری می‌کند، که می‌تواند شامل توزیع‌های پیوسته یا گسسته باشد.

مثال: برای همان 3 پارامتر، ممکن است صرف نظر از کل ترکیب‌های ممکن، فقط 20 ترکیب تصادفی را آزمایش کنید.

مزایا: برای فضاهای جستجوی بزرگ بسیار سریع‌تر است، اغلب پارامترهای نزدیک به بهینه را با تعداد آزمایش‌های کمتر پیدا می‌کند و توزیع پارامترهای انعطاف‌پذیر را امکان‌پذیر می‌سازد.

معایب: ممکن است بهترین ترکیب مطلق را از دست بدهد زیرا همه احتمالات را ارزیابی نمی‌کند.

تفاوت کلیدی: جستجوی شبکه‌ای جامع است و هر ترکیبی را آزمایش می‌کند، در حالی که جستجوی تصادفی به طور تصادفی زیرمجموعه‌ای از ترکیب‌ها را نمونه‌برداری می‌کند و آن را برای فضاهای ابرپارامتری بزرگ یا پیچیده کارآمدتر می‌کند.

## سوال ۲۱ : چه چیزی بهینه‌سازی بیزی را هوشمند می‌کند؟

بهینه‌سازی بیزی هوشمند در نظر گرفته می‌شود زیرا هنگام جستجوی ابرپارامترهای بهینه، به طور هوشمندانه بین اکتشاف (امتحان مناطق جدید) و بهره‌برداری (تمرکز بر مناطق شناخته شده) تعادل برقرار می‌کند.

چرا هوشمند است:

۱. از یک مدل احتمالی استفاده می‌کند
۲. یک مدل جایگزین (معمولًاً فرآیند گاووسی یا تخمین گر پارزن ساختار درختی) از تابع هدف (مثلاً امتیاز F1) می‌سازد که هر دو مورد زیر را پیش‌بینی می‌کند:

مقدار مورد انتظار امتیاز

عدم قطعیت آن پیش‌بینی

این به آن اجازه می‌دهد تا در مورد محل جستجوی بعدی تصمیمات آگاهانه‌ای بگیرد.

تابع اکتساب : از یک تابع اکتساب (مانند بهبود مورد انتظار یا UCB) برای تصمیم‌گیری در مورد محل نمونه‌برداری بعدی استفاده می‌کند که این تابع مناطقی با امتیاز پیش‌بینی شده بالا را ترجیح می‌دهد و همچنین مناطقی با عدم قطعیت بالا را ترجیح می‌دهد و این بدء‌بستان چیزی است که آن را کارآمد می‌کند.

از گذشته می‌آموزد: هر نمونه جدید مدل را به روزرسانی می‌کند و تصمیمات آینده را هوشمندانه‌تر می‌کند و از اتلاف وقت برای آزمایش مناطقی که از قبل عملکرد ضعیفی دارند، جلوگیری می‌کند.

بهینه‌سازی بیزی هوشمندانه است زیرا آنچه را که نمی‌داند مدل‌سازی می‌کند و با یادگیری بیشتر، استراتژی خود را به روزرسانی می‌کند. این امر آن را بسیار سریع‌تر و کارآمدتر از امتحان کورکورانه ترکیبات می‌کند.

## سوال ۲۲ : فرمول‌های (Accuracy)، (Precision) و (Recall) و امتیاز F1 چیست؟

نسبت پیش‌بینی‌های صحیح (مثبت و منفی) : **Accuracy**

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

چند مورد از موارد مثبت پیش‌بینی‌شده واقعاً مثبت هستند؟ **Precision**

$$Precision = \frac{TP}{TP + FP}$$

چند مورد مثبت واقعی به درستی شناسایی شده‌اند؟ **Recall**

$$Recall = \frac{TP}{TP + FN}$$

میانگین هارمونیک دقت و فراخوانی (هر دو را متعادل می‌کند) : **F1-score**

$$F1-score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

## سوال ۲۳ : منحنی ROC و منحنی AUC چیستند؟

منحنی **ROC** یک نمودار گرافیکی است که عملکرد یک طبقه‌بندی‌کننده دودویی را با رسم نمودار نرخ مثبت واقعی (TPR) (که حساسیت یا فراخوانی نیز نامیده می‌شود) (کسری از نمونه‌های مثبت که به درستی طبقه‌بندی شده‌اند) در برابر نرخ مثبت کاذب (FPR) (کسری از نمونه‌های منفی که به اشتباه به عنوان مثبت طبقه‌بندی شده‌اند) در آستانه‌های طبقه‌بندی مختلف نشان می‌دهد.

$$TPR = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$FPR = \frac{False\ Positives}{False\ Positives + True\ Negatives}$$

طبقه‌بندی کننده برای هر نمونه یک احتمال یا امتیاز ارائه می‌دهد. با تغییر آستانه برای طبقه‌بندی یک نمونه به عنوان مثبت (مثلاً از ۰ تا ۱)، مقادیر TPR و FPR متفاوتی به دست می‌آورید. هر نقطه روی منحنی ROC نشان‌دهنده یک جفت (TPR, FPR) برای یک آستانه خاص است.

منحنی از (۰،۰) (بدون پیش‌بینی مثبت) شروع می‌شود و در (۱،۱) (همه پیش‌بینی‌های مثبت) پایان می‌یابد. منحنی نزدیک‌تر به گوشه بالا سمت چپ نشان‌دهنده عملکرد بهتر است، زیرا TPR بالا را با FPR پایین نشان می‌دهد. یک خط مورب (از (۰،۰) تا (۱،۱)) نشان‌دهنده یک طبقه‌بندی کننده تصادفی (بدون قدرت تمایز) است.

**AUC (مساحت زیر منحنی ROC)**: AUC یک مقدار اسکالار واحد است که عملکرد کلی طبقه‌بندی کننده را خلاصه می‌کند. این مقدار نشان‌دهنده مساحت زیر منحنی ROC است که از ۰ تا ۱ متغیر است.

**AUC = 1** : طبقه‌بندی کننده کامل (جداسازی کامل کلاس‌ها).

**AUC = 0.5** : طبقه‌بندی کننده تصادفی (بهتر از حدس زدن نیست).

**AUC < 0.5** : بدتر از تصادفی (به ندرت، نشان می‌دهد که مدل به طور سیستماتیک اشتباه است).  
مقادیر بالاتر AUC نشان دهنده عملکرد بهتر مدل در تمام آستانه‌ها است.

## سوال ۲۴ Confusion Matrix چیست ؟

Confusion Matrix جدولی است که برای ارزیابی عملکرد یک مدل طبقه‌بندی، به ویژه برای وظایف طبقه‌بندی دودویی یا چند کلاسی، استفاده می‌شود. این ماتریس تعداد پیش‌بینی‌های صحیح و نادرست را در مقایسه با برچسب‌های واقعی نشان می‌دهد.

	Predicted Positive	Predicted Negative
Actual Positive	False Negative (FN)	True Positive (TP)
Actual Negative	True Negative (TN)	False Positive (FP)

تعاریف:

(TP) True Positive: مدل به درستی مثبت پیش‌بینی می‌کند.

(TN) True Negative: مدل به درستی منفی پیش‌بینی می‌کند.

(FP) False Positive: مدل مثبت پیش‌بینی می‌کند، اما در واقع منفی است (خطای نوع اول)

(FN) False Negative: مدل منفی پیش‌بینی می‌کند، اما در واقع مثبت است (خطای نوع دوم)

## فصل چهارم تحلیل داده ها در EDA

هدف از تحلیل اکتشافی داده ها (EDA) شناسایی الگوهای بررسی توزیع ویژگی ها، کشف مقادیر پرت (outliers)، بررسی مقادیر گمشده، و تحلیل رابطه میان ویژگی ها و متغیر هدف (درآمد) است. در این پروژه، تحلیل EDA شامل بخش های زیر بوده است:

### 1. بررسی اجمالی داده ها

در ابتدا، چند سطر ابتدایی از داده ها با استفاده از تابع head() نمایش داده شد تا ساختار کلی داده ها بررسی شود. همچنین با استفاده از توابع info() تعداد ردیفها، نوع داده ها و میزان کامل بودن هر ستون بررسی گردید

### 2. بررسی مقادیر گمشده و نامعتبر

برخی از ستون ها مانند native-country, occupation, workclass دارای مقداری نبودند

### 3. تحلیل آماری ویژگی های عددی

ویژگی های عددی مانند capital-gain, capital-loss, hours-per-week, education-num, age و بیشینه قرار گرفتند:

از تابع describe() برای بررسی میانگین، میانه، انحراف معیار، کمینه و بیشینه استفاده شد.

توزیع این ویژگی ها با استفاده از هیستوگرام و boxplot ترسیم شد.

### 4. تحلیل ویژگی های طبقه ای

ویژگی هایی مانند sex, race, marital-status, occupation, workclass, education و native- country، که داده هایی طبقه ای (categorical) دارند، با استفاده از نمودارهای میله ای تحلیل شدند و

متوجه شدیم که بیشتر افراد دارای تحصیلات Some-college یا HS-grad بودند و بیشترین کلاس شغلی مربوط به Private است و نژاد غالب White و جنسیت غالب Male است.

#### 5. بررسی رابطه ویژگی‌ها با متغیر هدف (درآمد)

یکی از مهم‌ترین بخش‌های EDA بررسی ارتباط بین ویژگی‌ها و برچسب درآمد (income) بود. برای این منظور نمودار میله‌ای (countplot) برای مقایسه توزیع income در بین دسته‌های مختلف رسم شد که نتایج نشان می‌داد که با افزایش سن، احتمال درآمد بالای ۵۰ هزار دلار بیشتر می‌شود و افراد دارای شغل‌های مدیریتی و حرفه‌ای، احتمال بیشتری برای درآمد بالا دارند و همچنین مردان بیشتر از زنان در دسته درآمد بالا قرار دارند.

#### 6. بررسی همبستگی ویژگی‌های عددی

برای بررسی میزان وابستگی متقابل بین ویژگی‌های عددی از نمودار Heatmap استفاده شد

## فصل پنجم توضیحات درباره پیش‌پردازش‌های انجام شده

پیش‌پردازش داده‌ها (Data Preprocessing) یکی از مراحل اولیه و بسیار مهم در پروژه‌های یادگیری ماشین است که هدف آن آماده‌سازی داده‌ها برای تحلیل یا آموزش مدل است. داده‌های خام معمولاً دارای نویز، مقادیر گمشده، فرمتهای ناهمانگ یا ویژگی‌های غیرضروری هستند. در نتیجه، پیش‌پردازش کمک می‌کند تا کیفیت داده‌ها بهبود یابد و مدل‌ها عملکرد بهتری داشته باشند و در این پروژه پیش‌پردازش داده‌ها شامل موارد زیر بود:

۱. حذف یا جایگزینی مقادیر گمشده: داده‌ای که مقادیر occupation و workclass، occupation آن‌ها خالی بود با عبارت "Missing" جایگذاری شدند.

```
1 # Fill the null values of occupation , workclass and native_country columns with a new category like "Missing"  
2 df_train['occupation'] = df_train['occupation'].fillna('Missing')  
3 df_train['workclass'] = df_train['workclass'].fillna('Missing')  
4 df_train['native-country'] = df_train['native-country'].fillna('Missing')
```

۲. تمیز کردن برچسب‌های درآمد: حذف نقطه . در انتهای برچسب‌های دیتاست آزمون.

```
1 print("Train labels:", df_train['income'].unique())  
2 print("Test labels:", df_test['income'].unique())  
  
Train labels: ['<=50K' '>50K']  
Test labels: ['<=50K' '>50K']
```

### ۳. کدگذاری داده‌های دسته‌ای:

دستون (Column)	تعداد مقادیر یکتا (Unique Values)	بهترین روش کدگذاری (Best Encoding)	دلیل انتخاب (Why)
workclass	حدود ۹	One-Hot Encoding	تعداد کم و دسته‌بندی‌های مستقل از هم
education	حدود ۱۶	Label Encoding	اگر ترتیب تحصیلی حفظ شود، لیبل بهتر است؛ در غیر این صورت One-Hot
marital-status	حدود ۷	One-Hot Encoding	دسته‌بندی‌های مجزا و بدون ترتیب مشخص
occupation	حدود ۱۵	Label Encoding	تعداد متوسط؛ استفاده از لیبل برای صرفه‌جویی در حافظه
relationship	حدود ۶	One-Hot Encoding	دسته‌های کم؛ مناسب و ساده Hot
race	حدود ۵	One-Hot Encoding	دسته‌های بسیار کم
sex	۲	Binary Encoding (1/0)	ساده‌ترین حالت؛ فقط دو مقدار

One-Hot تعداد زیاد؛ باعث افزایش زیاد ابعاد می شود	Label Encoding	حدود ۴۲	native-country
متغیر هدف (target) که از قبل دودویی است	Binary Encoding (1/0)	۲	income

۴. مقیاس‌بندی ویژگی‌های عددی: تمام ویژگی‌هایی عددی مانند age, hours-per-week استاندارد شدند. StandardScaler

```
2 from sklearn.preprocessing import StandardScaler
3 scaler = StandardScaler()
4 df_train[num_cols] = scaler.fit_transform(df_train[num_cols])
5 df_test[num_cols] = scaler.transform(df_test[num_cols])
```

## فصل ششم عملکرد مدل های استفاده شده

در ادامه عملکرد مدل های پر کاربرد در مسأله‌ی طبقه‌بندی آورده شده است:

### K-Nearest Neighbors (KNN) .1

ایده اصلی: یک نمونه جدید را به کلاس اکثربیت  $k$  همسایه نزدیک آن (بر اساس فاصله مانند اقلیدسی یا منهتن) نسبت می‌دهد.

کاربرد مناسب: زمانی که مرز بین کلاس‌ها پیچیده ولی داده کم است.

چند نکته‌ی مهم درباره‌ی این مدل :

- حساس به مقیاس داده‌ها : نرمال‌سازی ضروری است.
- به شدت وابسته به انتخاب  $k$ .
- پیش‌بینی کند است چون باید برای هر نمونه، فاصله با همه‌ی نقاط آموزش محاسبه شود.

### Logistic Regression .2

ایده اصلی: یک مدل خطی است که باتابع سیگموید احتمال تعلق به کلاس را پیش‌بینی می‌کند.

کاربرد مناسب: زمانی که داده‌ها خطی قابل تفکیک باشند یا برای پایه‌گذاری اولیه تحلیل.

چند نکته‌ی مهم درباره‌ی این مدل :

- تفسیرپذیر بودن (ضریب ویژگی‌ها قابل تحلیل است).
- نیاز به فرض‌هایی مثل مستقل بودن ویژگی‌ها.
- به داده پرت (outlier) حساس است.

### Decision Tree .3

ایده اصلی: ساختن یک درخت با سوال هایی از نوع "آیا  $x > a$ " برای تقسیم بندی نمونه ها به کلاس ها.

کاربرد مناسب: زمانی که به مدل تفسیر پذیر نیاز دارد.

چند نکته ی مهم درباره ی این مدل :

- اگر عمق درخت زیاد شود overfitting رخ می دهد.
- به راحتی می تواند با داده های طبقه بندی یا عددی کار کند.
- نیاز به نرم افزاری خاصی ندارد.

### Random Forest .4

ایده اصلی: ساخت چندین درخت تصمیم روی زیر مجموعه های داده و گرفتن میانگین یا رأی اکثریت.

کاربرد مناسب: زمانی که دقت بالا و مقاومت در برابر overfitting مهم باشد

چند نکته ی مهم درباره ی این مدل :

- نسبت به درخت تکی دقیق تر و پایدار تر است.
- کندر است ولی موازی سازی آن ساده است.
- ویژگی های مهم را می تواند مشخص کند (feature importance).

## LightGBM .5

ایده اصلی: نوعی از Gradient Boosting است که از تکنیک هایی مثل leaf-wise growth برای افزایش سرعت و کاهش مصرف حافظه استفاده می کند.

کاربرد مناسب: داده های بسیار بزرگ، ویژگی های زیاد.

چند نکته ی مهم درباره ی این مدل :

بسیار سریع تر از XGBoost و مناسب داده های sparse (مثلاً داده های متنه).

تنظیم پارامترها کمی سخت است؛ ممکن است با داده های overfit شود.

به داده پرت حساس است.

## XGBoost (Extreme Gradient Boosting) .6

ایده اصلی: ساخت درخت های ضعیف به صورت تدریجی برای کاهش خطا و استفاده از تکنیک های overfitting منظم سازی برای جلوگیری از

چند نکته ی مهم درباره ی این مدل :

- نسبت به Random Forest کنترولی دقت بیشتر.
- قابلیت کنترل overfitting از طریق پارامترهای gamma, lambda, alpha
- پشتیبانی از cross-validation و early stopping داخلی.

## SVM (Support Vector Machine) .7

ایده اصلی: یافتن بهترین مرز تصمیم‌گیری (hyperplane) بین کلاس‌ها که بیشترین فاصله را با نمونه‌های دو کلاس دارد.

کاربرد مناسب: داده‌های با ابعاد بالا، مرزهای مشخص.

چند نکته‌ی مهم درباره‌ی این مدل :

- با kernel trick می‌تواند داده‌های غیرخطی را هم مدل کند.
- در داده‌های زیاد **کند** می‌شود.
- نیازمند نرمال‌سازی دقیق ویژگی‌هاست.

## MLP (Multilayer Perceptron) .8

ایده اصلی: یک شبکه عصبی با چند لایه (مخفي) که با وزن‌دهی غیرخطی یاد می‌گيرد.

کاربرد مناسب: روابط پیچیده و غیرخطی بین ویژگی‌ها.

چند نکته‌ی مهم درباره‌ی این مدل :

- قدرت مدل‌سازی بالا ولی نیازمند داده زیاد و زمان آموزش بیشتر.
- به شدت به تنظیم پارامترها وابسته است: تعداد لایه‌ها، نرون‌ها، نرخ یادگیری، regularization و ...
- نیاز به نرمال‌سازی داده‌ها برای عملکرد مناسب.

## فصل هفتم مقایسه نتایج به دست آمده

در یک پژوهشی طبقه‌بندی داده‌های ساختاریافته (پیش‌بینی درآمد افراد بر اساس ویژگی‌های جمعیت‌سناختی مانند سن، تحصیلات، شغل و ...)، مجموعه‌ای از مدل‌های یادگیری ماشین با رویکردهای مختلف مورد بررسی و ارزیابی قرار گرفته‌اند. هدف از این تحلیل، مقایسه‌ی جامع عملکرد مدل‌ها بر اساس معیارهای استاندارد در زمینه‌ی طبقه‌بندی است. که برای مثال می‌توان به معیارهای زیر اشاره کرد :

دقت (Accuracy) : درصد نمونه‌هایی که به درستی طبقه‌بندی شده‌اند.

Precision : درصد پیش‌بینی‌های مثبت که واقعاً مثبت بوده‌اند.

Recall : توانایی مدل در یافتن تمام موارد مثبت واقعی.

F1-score : میانگینی موزون از Precision و Recall که برای عملکرد مدل‌های است، مخصوصاً زمانی که داده‌ها نامتوازن باشند.

در این تحلیل، انواع مختلفی از مدل‌های محبوب یادگیری ماشین شامل:

Decision Tree و Logistic Regression

مدل‌های مبتنی بر درخت‌های ترکیبی: مثل LightGBM، XGBoost، Random Forest

مدل‌های غیرخطی: مانند MLP و SVM

الگوریتم‌های ساده‌تر: مانند K-Nearest Neighbors (KNN)

در سناریوهای مختلف (عادی، Bayesian، Randomized Search، Grid Search، Optimization) اجرا و ارزیابی شده‌اند.

در ادامه تلاش می‌کنیم تا با بررسی دقیق نتایج عددی بهدست آمده از هر مدل بهترین مدل را از نظر کلی انتخاب کنیم ، و بسته به شرایط مختلف، پیشنهادهایی برای استفاده از مدل مناسب ارائه دهد. همچنین بررسی می‌شود که چگونه روش‌های تنظیم ابرپارامترها (Hyperparameter Tuning) مانند Bayesian Optimization یا Randomized Search یا Grid Search گذاشته‌اند.

### جدول نتایج معیار‌های گفته شده برای هر مدل :

	Model	Train-Type	Accuracy	Precision	Recall	F1-score
0	XGBoost-Bayesian	Bayesian-Optimization	0.859100	0.683192	0.752470	0.716159
1	LightGBM-Randomized	Randomized-Search	0.852527	0.660377	0.773531	0.712490
2	XGBoost-Grid	Grid-Search	0.846570	0.640769	0.797712	0.710679
3	XGBoost	Normal	0.854800	0.671369	0.754810	0.710649
4	LightGBM	Normal	0.856643	0.679572	0.743890	0.710278
5	LightGBM-Grid	Grid-Search	0.842147	0.627804	0.814873	0.709210
6	XGBoost-Randomized	Randomized-Search	0.854739	0.682885	0.718929	0.700443
7	RF-Randomized	Randomized-Search	0.825195	0.593563	0.824753	0.690316
8	RF	Normal	0.840919	0.651398	0.702548	0.676007
9	RF-Grid	Grid-Search	0.800197	0.548742	0.867915	0.672374
10	MLP	Normal	0.819667	0.588971	0.783151	0.672321
11	DT-Grid	Grid-Search	0.804803	0.558067	0.834633	0.668889
12	DT-Randomized	Randomized-Search	0.813648	0.576633	0.794332	0.668198
13	Logistic-Randomized	Randomized-Search	0.805602	0.561830	0.804472	0.661606
14	Logistic-Grid	Grid-Search	0.804557	0.559820	0.807852	0.661345
15	Logistic	Normal	0.801179	0.553713	0.816173	0.659800
16	SVM-Grid	Grid-Search	0.787974	0.532465	0.840094	0.651806
17	SVM	Normal	0.761563	0.497358	0.881175	0.635835
18	KNN	Normal	0.786991	0.533546	0.781591	0.634177
19	KNN-Grid	Grid-Search	0.789018	0.540334	0.715809	0.615815
20	DT	Normal	0.801486	0.572133	0.633125	0.601086
21	KNN-Randomized	Randomized-Search	0.787544	0.543927	0.622985	0.580778

حال با توجه به جدول نتایج ، بهترین مدل ها بر اساس معیار F1-Score عبارتند از:

ردیف	مدل	F1-score
1	XGBoost-Bayesian	0.7161
2	LightGBM-Randomized	0.7125
3	XGBoost-Grid	0.7107
4	XGBoost	0.7106
5	LightGBM	0.7102

### مقایسه دسته‌ای مدل‌ها

#### (6، 3، 2، 0) (مدل‌های XGBoost ◆)

این مدل عملکرد قوی در همه معیارها دارد و بهترین مدل از نظر Accuracy و F1-score است و مدل بهینه سازی شده‌ی XGBoost با استفاده از Grid Search دقیق‌تر است. خوبی دارد ولی F1 پایین‌تر دارد.

#### (5، 4، 1، 0) (مدل‌های LightGBM ◆)

این مدل عملکردی نزدیک به XGBoost دارد اما سرعت یادگیری و اجرای بالاتری دارد. مدل Normal بهتر از Randomized-Search و Grid است. Precision F1 بالایی دارد ولی Precision XGBoost کمتر از دارد.

#### (9، 8، 7، 6) (مدل‌های Random Forest ◆)

دقیق‌تر کلی قابل قبول است اما Precision XGBoost/LightGBM پایین‌تر از Precision Grid Search است. Precision Recall بسیار بالا (0.8679) است ولی Precision پایین دارد (0.5487).

## MLP ◆

عملکرد متوسطی دارد و دقت  $F1=0.672$  است و دقت یا Accuracy پایین‌تر از مدل‌های Boosting دارد ولی بهتر از SVM و KNN عمل کرده است و همچنین برای بهتر شدن نیاز به تنظیمات بیشتر دارد

## (20، 12، 11) (مدل‌های Decision Tree ◆

در حالت Grid و Randomized عملکرد بهتری از حالت Normal دارند. ساده‌ترین مدل است ولی دقت کمتری نسبت به Boosting و Random Forest دارد

## (15، 14، 13) (مدل‌های Logistic Regression ◆

عملکرد پایدار و مناسبی دارد و دقتی در حدود 80٪ دارد و Precision پایین (حدود 0.56-0.55) دارد ولی Recall بالا (0.80+) دارد

## (17، 16) (مدل‌های SVM ◆

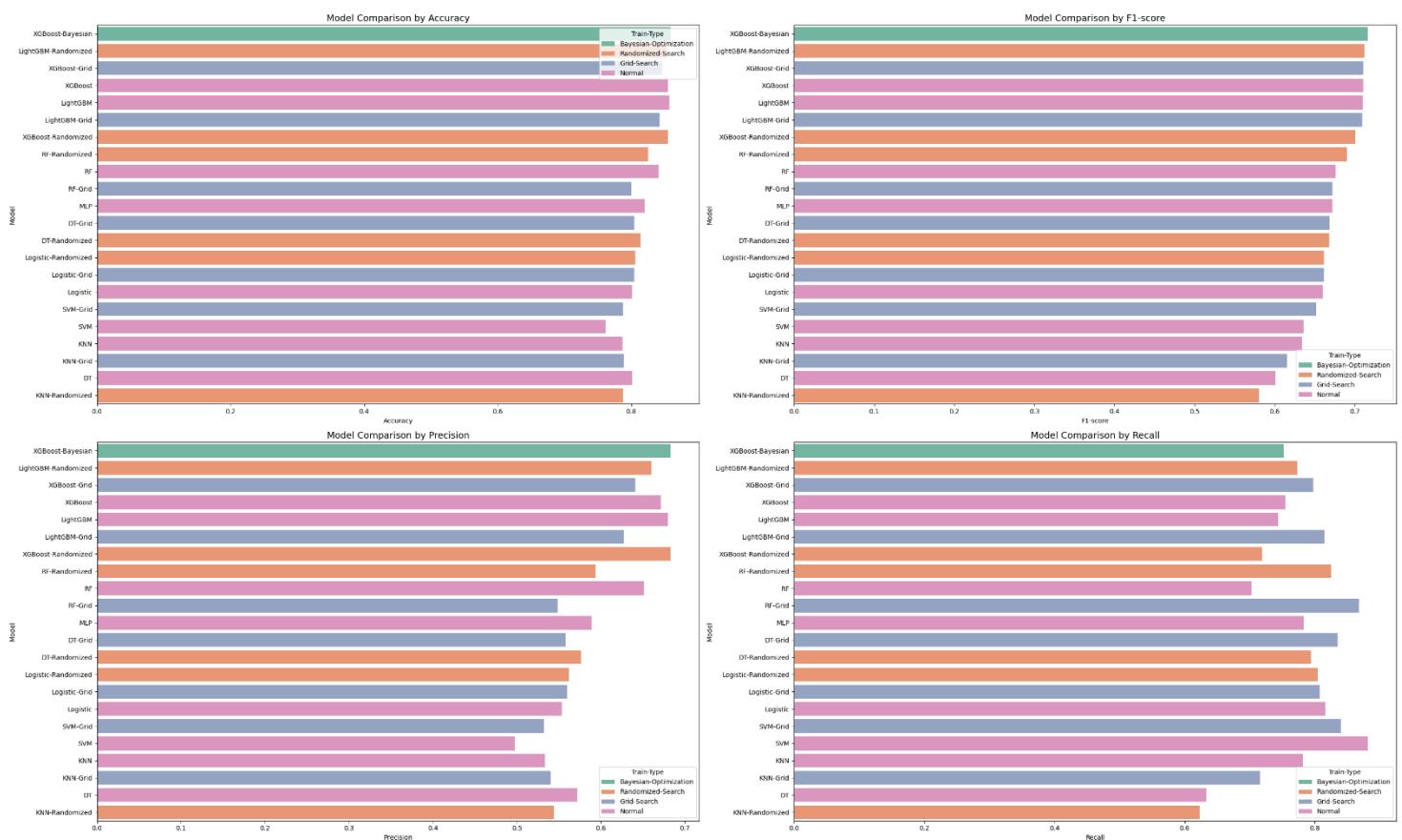
Precision بسیار بالا (تا 0.88) ولی Recall بسیار پایینی دارد

## (21، 19، 18) (مدل‌های KNN ◆

دقت و F1 پایین‌تر از سایر مدل‌ها دارد و استفاده از Grid Search برای بهینه سازی آن تفاوت چشمگیری ایجاد نکرده است

در ادامه نمودار برای مقایسه‌ی مدل‌های ۴ معیار گفته شده آورده شده است :

## فصل هفتم : مقایسه نتایج به دست آمده



## **منابع و مراجع**